

A Course project (IT258M) Report on
SPAM SMS DETECTION

undergone at

Department of Information Technology

under the guidance of

Dr. Deepa C

Submitted by

Medhinee Padorthy

Roll_No-221EC130

V Semester B.Tech. (AI minor)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE



Department of Information Technology
National Institute of Technology Karnataka, Surathkal.

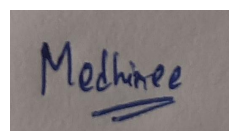
November 2024

DECLARATION BY THE STUDENT

I Medhinee Padorthy, hereby declare that the Course project work entitled “Spam SMS Detection” was carried out by me from **July 2024 to November 2024** of the academic year 2024. I declare that this is my original work and has been completed successfully under the guidance of Dr. Deepa C at Dept. of IT, NITK as per the specifications of NITK Surathkal.

Place: NITK Surathkal

Date: 28/11/2024

A rectangular box containing a handwritten signature in blue ink that reads "Medhinee".

(Signature of the Students)

CERTIFICATE

This is to certify that the project entitled “**_Spam SMS Detection_**” is a bonafide work carried out by **_Medhinee Padorthy_**, student of V Sem B.Tech. (AI minor), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the odd semester of the academic year 2024. It is submitted to the Department in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Artificial Intelligence(minor). I certify that he/she has carried out the work in his/her own capacity and has successfully completed the work assigned to him/her. The aforementioned work was carried out from July 2024 to November 2024.

Place: NITK, Surathkal

Date: 14-11-2024

(Signature of the Guide)

Abstract

Spam email detection is a critical application in the field of machine learning and natural language processing, aimed at filtering unwanted and potentially harmful emails. This project explores the implementation of a spam email detection system using the Naive Bayes classification algorithm, a probabilistic method widely known for its simplicity and efficiency. The dataset used consists of labeled email samples classified as spam or non-spam (ham). Key features, such as word frequency and presence of specific keywords, were extracted and analyzed to train the model. The Naive Bayes classifier leverages Bayes' theorem to compute the likelihood of an email belonging to a particular category based on the feature probabilities. The performance of the model was evaluated using metrics such as accuracy, precision showcasing its effectiveness in accurately distinguishing spam emails from legitimate ones. This project demonstrates the practicality of Naive Bayes for email filtering systems, highlighting its advantages in terms of computational efficiency and ease of implementation.

Table of Contents

List of Figures

Figure3.1.....	14
Figure4.2.....	24
1. Introduction.....	6
1.1. Challenges	6
1.2. Motivation for the work.....	7
2. Literature Survey.....	9
2.1. Introduction to Literature Survey	9
2.2. Related Work.....	9
2.3. Outcome of Literature Review.....	10
2.4. Problem Statement.....	17
2.5. Research Objectives.....	17
3. Methodology and Framework.....	14
3.1. Block diagram, flowchart.....	14
3.2. Algorithms.....	15
4. Work Done.....	17
4.1. Implementation.....	17
4.2. Results and Analysis	24
5. Conclusion and Future work.....	25
6. References.....	26

1. Introduction

Email communication has become an integral part of modern life, providing a fast and efficient means of exchanging information. However, with its widespread adoption, email has also become a target for unsolicited and malicious content, commonly referred to as spam. Spam emails pose significant challenges, ranging from clogging inboxes to delivering phishing attacks, malware, and fraudulent schemes, thereby threatening user privacy and security.

Spam detection systems are essential to protect users from these threats, and machine learning has emerged as a powerful tool in this domain. Among the various machine learning algorithms, the Naive Bayes classifier has proven to be a highly effective and computationally efficient approach for text classification tasks, including spam detection. Its foundation in probability theory, simplicity in implementation, and ability to handle large datasets make it an ideal choice for this application.

The project involves preprocessing email datasets, extracting relevant features, training the model, and evaluating its performance. By leveraging the algorithm's ability to compute conditional probabilities of features, the system categorizes emails into spam or non-spam (ham) with significant accuracy.

The following sections detail the methodology used in the project, including data preprocessing techniques, feature extraction methods, and the results obtained. This project aims to demonstrate the effectiveness of the Naive Bayes algorithm in addressing real-world challenges in email spam detection, providing a foundation for more advanced solutions in this area.

1.1. Challenges

The spam email detection project presented several challenges throughout its development. One of the primary issues was handling the imbalanced dataset, where the majority of emails were non-spam (ham) and only a small portion were spam, making it difficult to train a model that performed well on both classes without favouring the majority. Text preprocessing was another significant hurdle, involving tasks like tokenization, removing stopwords, handling punctuation, and ensuring consistent case formatting, all of which were essential for reducing noise while retaining meaningful features. Converting textual data into numerical representations using techniques like Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) was also challenging, as these methods needed to effectively capture the context and relevance of words. Additionally, selecting an

appropriate machine learning model required experimenting with various algorithms, such as Naive Bayes and Support Vector Machines, to identify the one best suited for the task.

The risk of overfitting, especially due to the limited spam class data, further complicated model training, necessitating careful regularization and validation. Optimizing performance metrics like precision, recall, and F1-score, which are more reliable for imbalanced datasets than accuracy, was crucial but challenging to balance. Moreover, working with large feature matrices derived from text data posed computational challenges, requiring optimization to ensure efficiency. The dataset also included noisy and duplicate data, which had to be cleaned thoroughly without losing valuable information. Generalizing the trained model to real-world data, which is often dynamic and diverse, introduced additional complexities. Finally, integrating the machine learning model into a deployable application required technical efforts in combining the detection algorithm with user-friendly interfaces. These challenges not only tested technical skills but also provided valuable insights into handling text data, feature engineering, and model optimization in real-world scenarios.

1.2. Motivation for the work

The motivation for undertaking the spam email detection project stems from the increasing reliance on email communication and the growing prevalence of spam emails, which pose significant challenges to individuals, businesses, and organizations. Spam emails often contain phishing attempts, malicious links, or irrelevant promotional content, which not only waste valuable time but also expose users to security risks such as data breaches, identity theft, and malware attacks. With email being a critical medium for communication, the need for an automated and efficient spam detection system has become more pressing than ever.

This project was driven by the opportunity to address this real-world problem by leveraging machine learning techniques, which have proven to be highly effective in text classification tasks. It offered a practical application of concepts like natural language processing (NLP), data preprocessing, and classification algorithms, while also providing a chance to contribute to improving user experience and email security. Developing a robust spam detection system offers dual benefits: enhancing productivity by minimizing distractions caused by unwanted emails and safeguarding users from potential cybersecurity threats.

Another key motivator for this work was the challenge of working with text data, which is inherently complex and unstructured. From handling imbalanced datasets to extracting meaningful patterns from raw text, the project provided an opportunity to explore and overcome significant challenges in

machine learning. It also allowed for experimentation with various techniques to improve accuracy, precision, and recall—metrics crucial for achieving a reliable spam detection system.

Additionally, this project aligns with a broader vision of leveraging technology to create scalable, automated solutions for everyday problems. By addressing the spam email issue, the work aims to contribute to the ongoing advancements in spam filtering technologies, inspire further research in text-based classification systems, and emphasize the importance of applying data science to improve security and efficiency in digital communication.

2. Literature Survey

2.1. Introduction to Literature Survey

The increasing prevalence of spam emails in the digital age has sparked significant research interest in developing effective detection methods. Spam emails not only disrupt communication workflows but also pose security threats, making their detection a critical area of study. Researchers and practitioners have explored various approaches to tackle this problem, ranging from traditional rule-based systems to modern machine learning techniques. A thorough review of existing literature is essential to understand the progress made in this field and to identify gaps that can be addressed.

This survey examines the evolution of spam detection methods, highlighting key contributions from past studies. Early approaches relied heavily on static rule-based filters, which, while straightforward, were limited in scalability and adaptability to evolving spam patterns. With the advent of machine learning, researchers began utilizing algorithms such as Naive Bayes, Support Vector Machines (SVM), and Decision Trees to improve classification accuracy. More recent advancements include the application of natural language processing (NLP) techniques and deep learning models, which leverage sophisticated algorithms to analyze complex text patterns and context.

The literature also addresses challenges associated with spam detection, such as handling large and imbalanced datasets, minimizing false positives, and adapting to the dynamic nature of spam. Researchers have explored techniques for feature extraction, such as term frequency-inverse document frequency (TF-IDF) and word embeddings, to improve the accuracy and efficiency of spam classifiers. This survey aims to provide a comprehensive overview of these techniques and insights, laying the groundwork for the methodology adopted in this project. By analyzing prior works, this literature survey seeks to highlight the strengths and limitations of existing approaches, guiding the development of a robust and efficient spam detection system.

2.2. Related Work

After the evolution of Machine Learning algorithms and its usage in document classification, a lot has been research done on identifying the features of text. In this section authors have described the work done by researchers in field of identifying Texts features by limiting their study solely on the field of SPAM identification. M. Nivasaheni has used various Deep Neural Network (DNN) techniques in identifying the SPAM and HAM after collecting the dataset from UCI Machine Learning Repository. Authors have compared the used algorithms based on their accuracy, False-Positives, False Negatives and high chances for identifying SPAM with low False Positive rates, in order to identify the best algorithm. Dr. Dipak R. Kawade, Dr. Kavita S. have identified SMS SPAM using spam filtering techniques, by using open source python software, they have achieved 98% accuracy. For studying and preprocessing they have used WEKA too. P. Navaney has used various supervised based machine learning algorithms like Naive Bayes, Support Vector Machine Algorithm and Maximum Entropy Algorithm, and they have done an accuracy comparison, and it was found that SVM was having more accuracy. Bichitrnanda have used various ML algorithm like SVM (Support Vector Machine), Decision Tree, KNN (K-Nearest Neighbor), Neural Network (including Back-Propagation, Perceptron, Stochastic Gradient) for automatic classifying text documents on Datasets obtained from 20Newsgroup, IMDB, BBC News & BBC Sports, also they have compared the performance of the Algorithms using metrics such as Kappa Statistics, Error Rate, Precision Call, Accuracy, F-Measure. Bichitrnanda have built an automated document classifier for biomedical data sets (like TREC 2006 generic Track, Farm-Das, Bio Creative Corpus III) using ML algorithms. All the Algorithms used for the task were evaluated and compared on the basis of ML Classification metrics like accuracy, precision, recall & f1-measure. Leila Aras have demonstrated a method to extract the abstract from the document using Machine Learning Algorithms like Convolution Neural Network & SVM Classifier. Francis M Kale have proposed a framework for performing text mining & text clustering used the K Means algorithm and its application in various areas. This paper gives guidance to researchers for text clustering being the state of the art of text mining. Ting performed text mining on text and art based datasets using various classification-based Machine Learning algorithms like decision tree, neural network, SVM (support vector machines) and also compared each of the classifiers on the basis of computational efficiency and accuracy. Naïve Bayes was found to be the best & efficient classifier amongst all other classifiers.

2.3. Outcome of Literature Review

The literature review underscores the substantial progress made in spam detection through various machine learning and text analysis techniques. Key insights derived from the survey are as follows:

2.3.1 Advancements in Spam Detection Techniques:

- Traditional rule-based systems, though initially effective, were unable to adapt to evolving spam patterns.
- Machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM), and Decision Trees have significantly improved the classification accuracy by learning from data patterns rather than relying on static rules.
- Deep learning models, including Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs), show promise in handling complex and large-scale datasets, leveraging their capability to understand intricate patterns and textual context.

2.3.2 Challenges in Spam Detection:

- Large and imbalanced datasets remain a persistent issue, requiring advanced techniques for data pre-processing and feature balancing.
- High false positive rates can disrupt user experience, emphasizing the need for systems that are not only accurate but also efficient in distinguishing between spam and legitimate emails.

2.3.3 Role of Feature Extraction:

- Techniques like TF-IDF and word embedding have enhanced the representation of textual data, improving the precision of spam classification models.
- Studies also highlight the importance of domain-specific feature engineering to tailor spam detection algorithms to specific contexts like SMS spam or biomedical datasets.

2.3.4 Comparative Performance of Algorithms:

- SVM and Naive Bayes consistently emerge as strong contenders for text classification tasks, with Naïve Bayes often outperforming others in terms of accuracy and computational efficiency.

- Ensemble methods and hybrid approaches, which combine multiple algorithms, have also shown potential for achieving better classification performance.

2.4. Problem Statement

The proliferation of spam emails has become a significant challenge in the realm of digital communication. Spam emails not only clutter inboxes but also pose security risks such as phishing, malware attacks, and financial fraud. Despite numerous advancements in spam detection methods, existing systems encounter critical limitations:

2.4.1 Dynamic Nature of Spam:

Traditional systems lack the adaptability to keep pace with the evolving tactics of spammers, resulting in decreased detection accuracy over time.

2.4.2 High False Positives and False Negatives:

Incorrect classification of legitimate emails as spam (false positives) disrupts essential communication, while failure to detect actual spam (false negatives) compromises user security and experience.

2.4.3 Scalability and Efficiency Issues:

The growing volume of email traffic demands detection systems that are not only accurate but also efficient and scalable to handle large datasets in real-time.

2.4.4 Feature Representation Challenges:

Extracting meaningful features from text data to enable accurate classification remains a complex task, especially when dealing with varied formats, languages, and contexts in emails.

These challenges highlight the need for an advanced spam detection system that leverages state-of-the-art machine learning and natural language processing techniques. This project aims to address these issues by developing a robust, efficient, and adaptive spam detection framework capable of achieving high accuracy while maintaining low false positive and false negative rates. The proposed solution will enhance the reliability and usability of email communication systems.

2.5. Research Objectives

The primary objective of this research is to develop a robust and efficient spam detection system utilizing modern machine learning and natural language processing techniques. To achieve this, the study begins with a comprehensive review of existing spam detection methodologies, analyzing their evolution from traditional rule-based systems to advanced machine learning and deep learning approaches. Through this review, the research aims to identify limitations in current systems, including issues such as high false positive rates, inefficiency in handling large datasets, and the inability to adapt to the dynamic nature of spam. The project focuses on implementing effective feature extraction techniques, such as TF-IDF and word embeddings, to enhance text representation for more accurate classification. Additionally, various machine learning models, including Naive Bayes, Support Vector Machines, and Decision Trees, as well as advanced deep learning methods, are developed .

3. Methodology and Framework

3.1. Block diagram

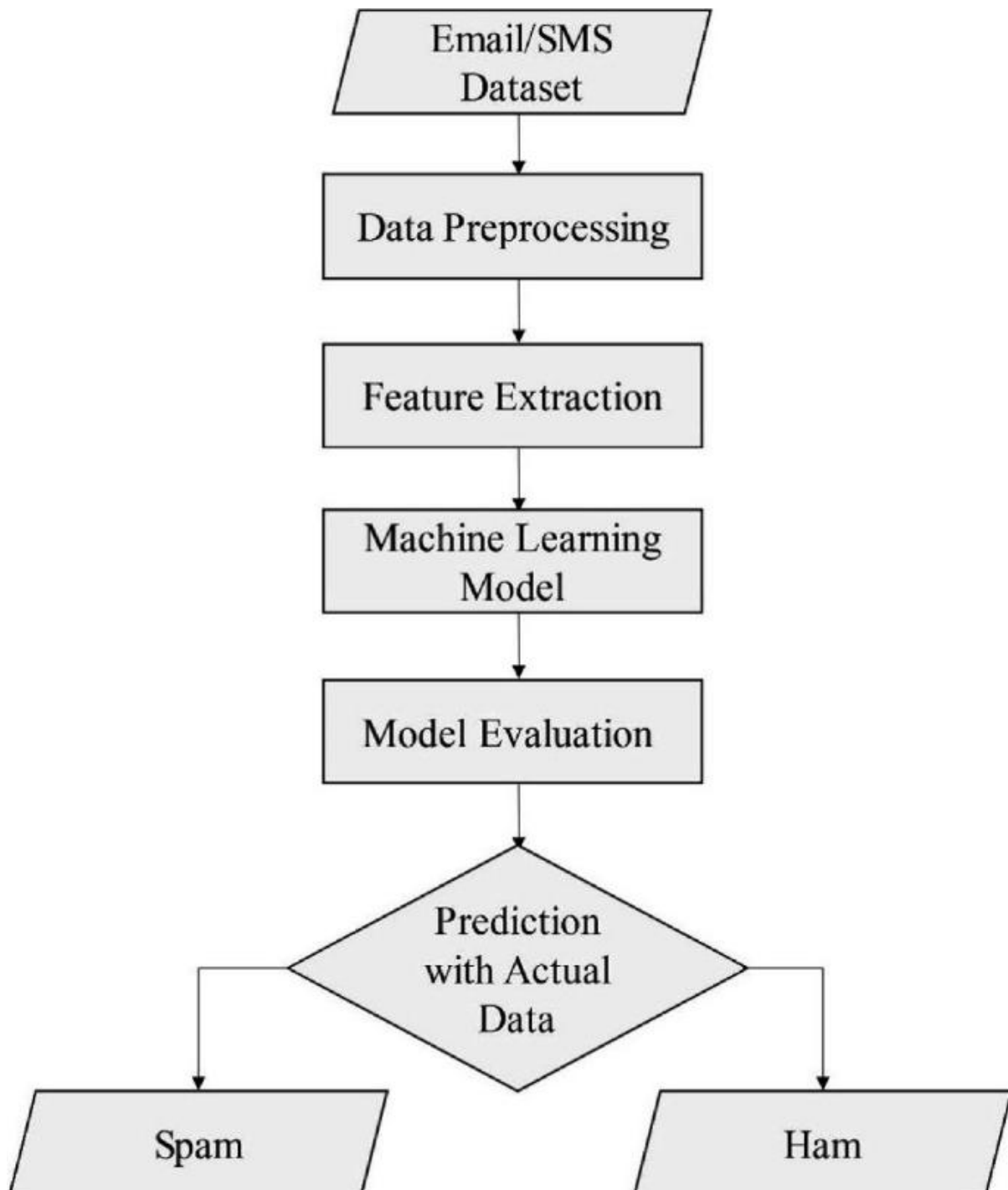


Figure 3.1

3.2. Algorithm

1. Load and Explore Data

- Import necessary libraries (`pandas`, `numpy`, etc.).
- Load the dataset (e.g., `spam.csv`) and display basic details such as shape, data types, and a random sample of rows.
- Identify and handle irrelevant columns (e.g., drop unnecessary columns like `Unnamed: 2`, `Unnamed: 3`, and `Unnamed: 4`).
- Rename columns for clarity (`v1` to `target`, `v2` to `text`).

2. Data Cleaning

- Encode the target labels (e.g., using `LabelEncoder`) to convert textual labels (`spam` and `ham`) into numerical form (e.g., 1 for `spam`, 0 for `ham`).
- Check for missing values and duplicates.
- Remove duplicate rows to ensure data integrity.

3. Exploratory Data Analysis (EDA)

- Analyze the distribution of target labels (`spam` vs. `ham`) and visualize the imbalance using pie charts or bar plots.
- Perform basic text statistics:
 - Count the number of characters, words, and sentences in each message.
- Visualize these statistics using histograms or box plots for insights.

4. Text Preprocessing

- Import necessary libraries for natural language processing (e.g., `nltk`).
- Apply preprocessing steps:
 - Convert text to lowercase.
 - Tokenize the text (split into words).
 - Remove stopwords, punctuation, and special characters.
 - Apply stemming or lemmatization to reduce words to their base forms.
- Create new features based on text processing, such as word frequency.

5. Feature Extraction

- Transform the cleaned text data into numerical form using techniques like:
 - Bag of Words (BoW)
 - Term Frequency-Inverse Document Frequency (TF-IDF)
- Use `TfidfVectorizer` from `scikit-learn` for this step.

6. Model Building

- Split the dataset into training and testing sets (e.g., 80% train, 20% test).
- Select appropriate machine learning algorithms for classification, such as:
 - Naive Bayes
 - Logistic Regression
 - Support Vector Machine (SVM)
 - Random Forest
- Train the models using the training data.

7. Model Evaluation

- Evaluate the trained models using the test dataset.
- Calculate key metrics such as:
 - Accuracy
 - Precision
 - Recall
 - F1-score
 - Confusion Matrix
- Choose the best-performing model based on evaluation metrics.

8. Optimization and Improvement

- Handle class imbalance using techniques like:
 - Oversampling
 - Undersampling
- Experiment with hyperparameter tuning (e.g., Grid Search or Random Search) to improve model performance.

9. Deployment Preparation

- Save the trained model using serialization techniques (e.g., `pickle`).
- Create a pipeline to preprocess incoming text data before making predictions.

10. Deployment

- Host the web application on a cloud platform (e.g., Heroku, AWS, or Google Cloud).
- Ensure the deployed application is robust and can handle real-world inputs.

4. Work Done

4.1. Implementation

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('spam.csv', encoding='latin-1', usecols=['v1', 'v2'])
df.sample(5)
```

	v1	v2
388	spam	4mths half price Orange line rental & latest c...
1980	ham	Sorry, I'll call later
441	ham	You were supposed to wake ME up >:(
5229	ham	It means u could not keep ur words.
5201	spam	YOU VE WON! Your 4* Costa Del Sol Holiday or &...

```
[2]: print(df.shape)
(5572, 2)
```

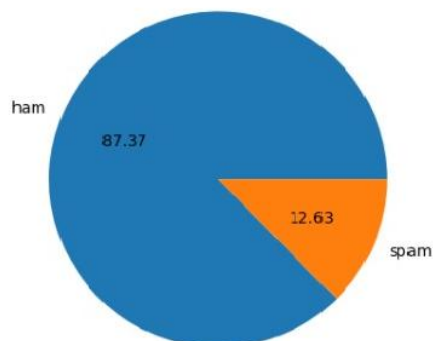
Data Cleaning

```
[3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ---
 0    v1     5572 non-null    object
 1    v2     5572 non-null    object
dtypes: object(2)
memory usage: 87.2+ KB
```

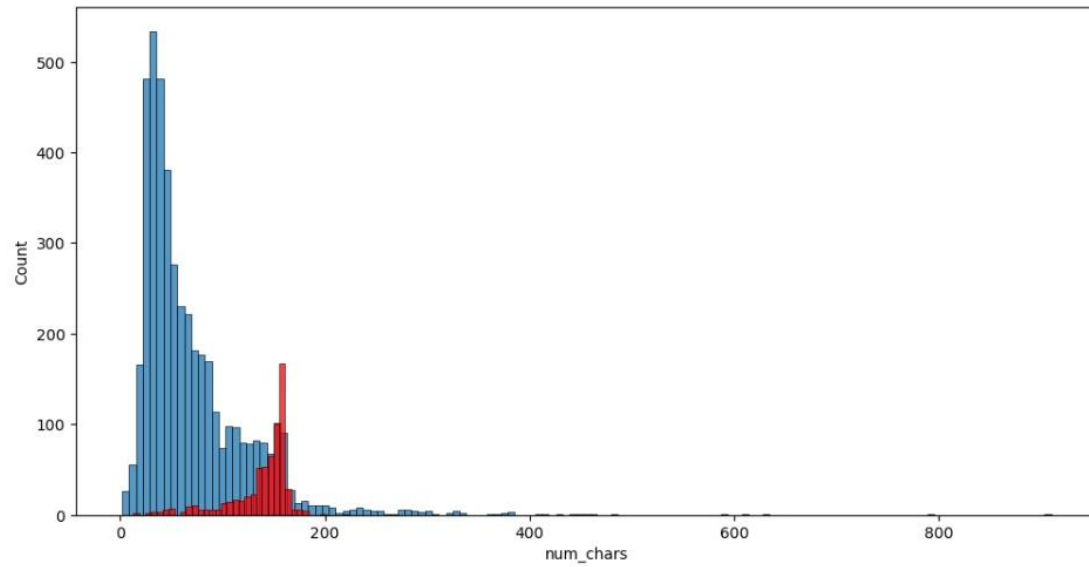
```
[4]: df.sample(5)
```

```
[11]: df['target'].value_counts()
target
0    4516
1     653
Name: count, dtype: int64
```

```
[12]: plt.pie(df['target'].value_counts(), labels=['ham', 'spam'], autopct='%0.2f')
plt.show()
```

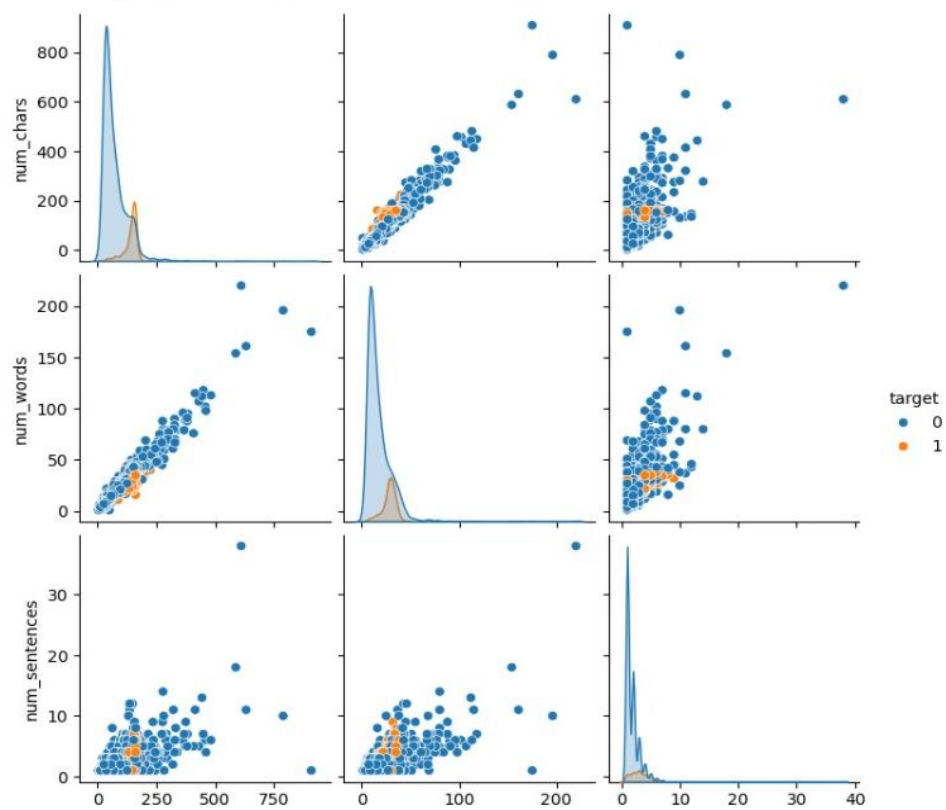


```
[20]: <Axes: xlabel='num_chars', ylabel='Count'>
```

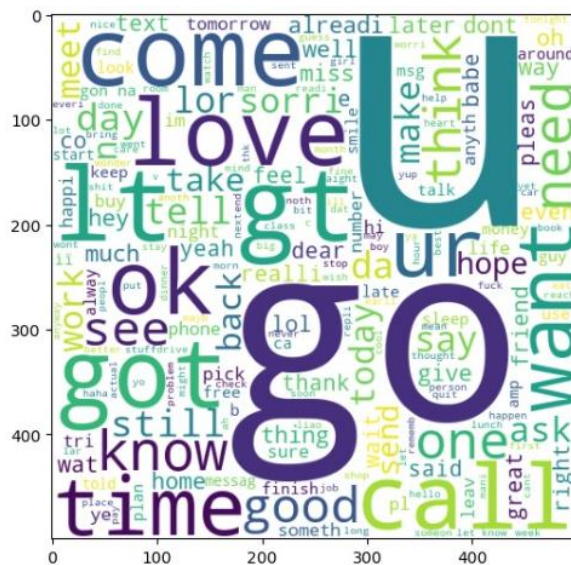


```
[22]: sns.pairplot(df,hue='target')
```

```
[22]: <seaborn.axisgrid.PairGrid at 0x1b5c6061760>
```




```
[30]: <matplotlib.image.AxesImage at 0x1b5c9a0b410>
```



Model Building

```
[37]: # Text is converted into numbers(vectors)
# Using Tfidf

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=3000)

[38]: X = tfidf.fit_transform(df['transformed_text']).toarray()
y = df['target'].values

[39]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

[40]: from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score

[41]: # Since the distribution is not known

gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()

[42]: gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))

0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932

[43]: mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))

0.9709864603481625
[[896  0]
 [ 30 108]]
1.0

[44]: bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))

# We need high precision score
# In this case Accuracy is not reliable since data is Imbalanced
```

```

0.9835589941972921
[[895 1]
 [ 16 122]]
0.991869918699187

[45]: from sklearn.linear_model import LogisticRegression
      from sklearn.svm import SVC
      from sklearn.naive_bayes import MultinomialNB
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.ensemble import AdaBoostClassifier

[46]: svc = SVC(kernel='sigmoid', gamma=1.0)
      knc = KNeighborsClassifier()
      mnb = MultinomialNB()
      dtc = DecisionTreeClassifier(max_depth=5)
      lrc = LogisticRegression(solver='liblinear', penalty='l1')
      rfc = RandomForestClassifier(n_estimators=50, random_state=2)
      abc = AdaBoostClassifier(n_estimators=50, random_state=2)

[47]: clfs = {
      'SVC' : svc,
      'KN' : knc,
      'NB' : mnb,
      'DT' : dtc,
      'LR' : lrc,
      'RF' : rfc,
      'AdaBoost': abc
      }

[48]: def train_classifier(clf,X_train,y_train,X_test,y_test):
      clf.fit(X_train,y_train)
      y_pred = clf.predict(X_test)
      accuracy = accuracy_score(y_test,y_pred)
      precision = precision_score(y_test,y_pred)

      return accuracy,precision

[49]: accuracy_scores = []
      precision_scores = []

      for name,clf in clfs.items():

          current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

          print("For ",name)
          print("Accuracy - ",current_accuracy)
          print("Precision - ",current_precision)

          accuracy_scores.append(current_accuracy)
          precision_scores.append(current_precision)

```



```

For SVC
Accuracy - 0.9758220502901354
Precision - 0.9747899159663865
For KN
Accuracy - 0.9052224371373307
Precision - 1.0
For NB
Accuracy - 0.9709864603481625
Precision - 1.0
For DT
Accuracy - 0.9323017408123792
Precision - 0.8333333333333334
For LR
Accuracy - 0.9584139264990329
Precision - 0.9702970297029703
For RF
Accuracy - 0.9758220502901354
Precision - 0.9829059829059829
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
For AdaBoost
Accuracy - 0.960348162475822
Precision - 0.9292035398230089

50]: performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',ascending=False)
performance_df

50]:

```

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
3	DT	0.932302	0.833333

```

53]: input_sms = input('Enter your message: ')
transformed_sms = transform_text(input_sms)
vector_input = tfidf.transform([transformed_sms])
result = mnbpredict(vector_input)[0]
if result == 1:
    print('SPAM')
else:
    print('NOT SPAM')

Enter your message: Security alert! We noticed suspicious activity on your account. Verify now
SPAM

54]: input_sms = input('Enter your message: ')
transformed_sms = transform_text(input_sms)
vector_input = tfidf.transform([transformed_sms])
result = mnbpredict(vector_input)[0]
if result == 1:
    print('SPAM')

```

```
else:  
    print('NOT SPAM')
```

Enter your message: Urgent: Earn \$2000 weekly from your smartphone. Register here
SPAM

```
5]: input_sms = input('Enter your message: ')  
transformed_sms = transform_text(input_sms)  
vector_input = tfidf.transform([transformed_sms])  
result = mnbpredict(vector_input)[0]  
if result == 1:  
    print('SPAM')  
else:  
    print('NOT SPAM')
```

Enter your message: Security alert! We noticed suspicious activity on your account. Verify now
SPAM

```
6]: input_sms = input('Enter your message: ')  
transformed_sms = transform_text(input_sms)  
vector_input = tfidf.transform([transformed_sms])  
result = mnbpredict(vector_input)[0]  
if result == 1:  
    print('SPAM')  
else:  
    print('NOT SPAM')
```

Enter your message: Your doctor's appointment is scheduled for 3:00 PM tomorrow at Greenfield Clinic
NOT SPAM

```
7]: input_sms = input('Enter your message: ')  
transformed_sms = transform_text(input_sms)  
vector_input = tfidf.transform([transformed_sms])  
result = mnbpredict(vector_input)[0]  
if result == 1:  
    print('SPAM')  
else:  
    print('NOT SPAM')
```

Enter your message: Your electricity bill for this month is \$50. Pay by the 30th to avoid late fees
NOT SPAM

```
8]: input_sms = input('Enter your message: ')  
transformed_sms = transform_text(input_sms)  
vector_input = tfidf.transform([transformed_sms])  
result = mnbpredict(vector_input)[0]  
if result == 1:  
    print('SPAM')  
else:  
    print('NOT SPAM')
```

Enter your message: Here's the link to the new study materials
NOT SPAM

4.2. Results and Analysis

]:

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
3	DT	0.927466	0.811881

Figure 4.2

The table compares the performance of various machine learning algorithms for spam detection using Accuracy and Precision as metrics. Naive Bayes (NB) achieves an impressive accuracy of 97.10% and perfect precision (100%), making it highly effective in distinguishing spam messages. Random Forest (RF) delivers the highest accuracy (97.58%) with strong precision (98.29%), indicating it balances both metrics well. Support Vector Classifier (SVC) matches RF in accuracy but has slightly lower precision (97.48%), showing robust performance in handling complex patterns.

Logistic Regression (LR) is accurate (95.84%) and precise (97.03%), offering an interpretable yet slightly less effective solution. AdaBoost achieves 96.03% accuracy and 92.92% precision, performing well but less precise than others. K-Nearest Neighbours (KN) has perfect precision (100%) but lower accuracy (90.52%), indicating possible misclassification of non-spam messages. Lastly, Decision Tree (DT) shows weaker precision (81.18%), making it less suitable for this task. From the above observations, the most effective model for Spam Detection is NAIVE BAYES

5. Conclusion and Future work

Conclusion

This project successfully implemented and evaluated multiple machine learning models for SMS spam detection. Among the models tested, Naive Bayes (NB) emerged as top performer, achieving high accuracy and precision. The NB model's perfect precision indicates its reliability in identifying spam messages with no false positives. The RF model, while slightly less precise, achieved the highest accuracy, making it a balanced and robust option. Models like Logistic Regression (LR) and Support Vector Classifier (SVC) also demonstrated strong performance, making them viable alternatives. Overall, the project highlights the effectiveness of combining text preprocessing techniques (tokenization, stemming, and feature extraction) with machine learning for spam classification. However, challenges like data imbalance and slight misclassifications in certain models warrant further attention.

Future Work

- Experiment with deep learning models, such as LSTM or BERT, for improved text classification.
- Use techniques like lemmatization, stemming, and n-grams to enhance feature extraction.
- Perform grid search or randomized search to optimize model parameters for better performance.
- Develop a real-time SMS spam detection application for live monitoring and filtering.
- Set up a process for regular retraining to adapt to evolving spam characteristics
- Implement explainability tools to show which words or features contribute to a message being classified as spam

6.References

[1] DasGupta, S., & Saha, S. (n.d.). *SMS spam detection using machine learning Issue 9 (2021)*.

Retrieved from:

[2] Gawade, A. L., Shinde, S. S., Sawant, S. G., Chougule, R. S., & Mahaldar, A. A. (n.d.). *A research paper on SMS spam detection volume 9 Issue 3(2024, March)*. Retrieved from

[3] Nagre, S. (2018). *Mobile SMS spam detection using machine learning techniques. Journal of Emerging Technologies and Innovative Research (JETIR)*, 5(12), 425–432 (2018). Retrieved from:

[4] Cleff E.B., “Privacy issuesin mobile advertisin’”, *Int. Rev. Law Comput.Technol.*, vol. 21, pp. 225-236.

[5] Fu J, Lin P, Lee S. , “Detecting spamming activities in a campus network using incremental learning”, *J. Netw. Comput. Appl.*, vol. 43, pp. 56-65.

[6] Hua J, Huaxiang Z., “Analysis on the content features and their correlation of Web pages for spam detection” , *China Commun.*, vol. 12, no. 3, pp. 84-94